AD-762 419

FORTRAN SUBROUTINES FOR BICUBIC SPLINE INTERPOLATION

John J. Cornyn

Naval Research Laboratory
Washington, D.C.

June 1973

NRL Memorandum Report 2596
NRL Computer Bulletin 32

Fortran Subroutines for Bicubic Spline Interpolation

JOHN J. CORNYN

Information Processing Systems Branch
Communications Sciences Division

June 1973

NAVAL RESEARCH LABORATORY
Washington, D.C.

47

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Research Laboratory Washington, D.C. 20375	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP ----

3. REPORT TITLE

FORTRAN SUBROUTINES FOR BICUBIC SPLINE INTERPOLATION

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

This is a final report on one phase of a continuing problem.

5. AUTHOR(S) (First name, middle initial, last name)

John J. Cornyn

6. REPORT DATE June 1973	7a. TOTAL NO. OF PAGES ~~46~~ 47	7b. NO. OF REFS 11

8a. CONTRACT OR GRANT NO. NRL General and Administrative b. PROJECT NO. Function S01-38 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) NRL Memorandum Report 2596 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) NRL Computer Bulletin 32

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Department of the Navy (Office of Naval Research) Washington, D.C. 20360

13. ABSTRACT

Two Fortran subroutines (BICUB1 and BICUB2) which perform bicubic spline interpolation of a tabulated function of two variables are described. Given the values X(1),...,X(N) and Y(1),...,Y(M) of two independent variables and the corresponding function values {U(I,J)=f(X(I), Y(J))}, I=1,...,N and J=1,...,M and certain normal derivatives (optional) along the boundaries of the x-y mesh, BICUB1 estimates the derivatives f_x, f_y, and f_{xy} at each (I,J) mesh point. If the normal derivatives along the mesh boundaries are unknown, BICUB1 estimates them using a moving third order two dimensional Lagrange interpolating polynomial. Given the coordinates (XPT,YPT) and the derivatives calculated by BICUB1, BICUB2 obtains the coefficients of the bicubic polynomial for the rectangular region of the mesh containing (XPT,YPT) and estimates the functional value UPT=f(XPT,YPT). In effect, the routines pass a twice continuously differentiable piecewise bicubic polynomial, $u(x,y) \in C^2$, through the given functional values.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Interpolation						
Spline						
Bicubic Spline						
Computer Program						
Hermite Polynomial						
Fortran						

CONTENTS

ABSTRACT

Two Fortran subroutines (BICUB1 and BICUB2) which perform bicubic spline interpolation of a tabulated function of two variables are described. Given the values X(1),...,X(N) and Y(1),...,Y(M) of two independent variables and the corresponding function values $\{U(I,J)=f(X(I), Y(J))\}$, I=1,...,N and J=1,...,M and certain normal derivatives (optional) along the boundaries of the x-y mesh, BICUB1 estimates the derivatives f_x, f_y, and f_{xy} at each (I,J) mesh point. If the normal derivatives along the mesh boundaries are unknown, BICUB1 estimates them using a moving third order two dimensional Lagrange interpolating polynomial. Given the coordinates (XPT,YPT) and the derivatives calculated by BICUB1, BICUB2 obtains the coefficents of the bicubic polynomial for the rectangular region of the mesh containing (XPT,YPT) and estimates the functional value UPT=f(XPT,YPT). In effect, the routines pass a twice continuously differentiable piecewise bicubic polynomial, $u(x,y) \in C^2$, through the given functional values.

PROBLEM STATUS

This is a final report on one phase of a continuing problem.

AUTHORIZATION

NRL Problem S01-38

1.0 IDENTIFICATION

1.1 Title

Bicubic Spline Interpolation

1.2 Identification Name

E1-NRL-BICUBIC

1.3 Classification Code

E1-Interpolation and Approximations, Curve Fitting

1.4 RCC Identification Number

E1001Ø00

1.5 Entry Points

BICUB1
BICUB2

1.6 Programming Language

Language: 3600/3800 FORTRAN

Routine type: Subroutines

Operating System: DRUM SCOPE 2.1

1.7 Computer and Configuration

CDC 3800

1.8 Contributor or Programmer

John J. Cornyn[1]
Information Processing Systems Branch (Code 5493)
Communications Sciences Division

---

[1]Formerly with the Large Aperture Systems Branch, Code 8160, Acoustics Division

1.9 Contributing Organization

NRL - Naval Research Laboratory,
Washington, D. C., 20375

1.10 Program Availability

1.10.1 Submittal: Program write-up, Fortran source deck, source listing

1.10.2 On File: RCC Program Library

1.11 Verification

Several third degree polynomials were used to test BICUBIC; answers were good to at least nine significant figures. Higher degree polynomials were also used. Then, as expected, the results did not compare as well with the true values.

1.12 Date

26 February 1973

2.0 PURPOSE

2.1 Description of Routines

Let the values u_{ij} of a function $u(x,y)$ over a two dimensional domain be given at the mesh points (x_i, y_j) where $i = 1, \ldots, N$; $j = 1, \ldots, M$.

(1) The first problem considered is the estimation of the normal derivatives along the boundaries of the mesh assuming they are unknown. In Figure 1, squares designate locations at which one needs to know the normal derivatives with respect to x, $p_{ij} = u_x(x_i, y_j)$. Circles designate locations at which one needs to know the normal derivatives with respect to y, $q_{ij} = u_y(x_i, y_j)$. Squares imbedded in circles designate locations where the normal derivatives with respect to both x and y, $S_{ij} = u_{xy}(x_i, y_j)$, are required, in addition to p_{ij} and q_{ij}. A solution to this problem will be given in the form of subroutines EDGES and LAGRAN.

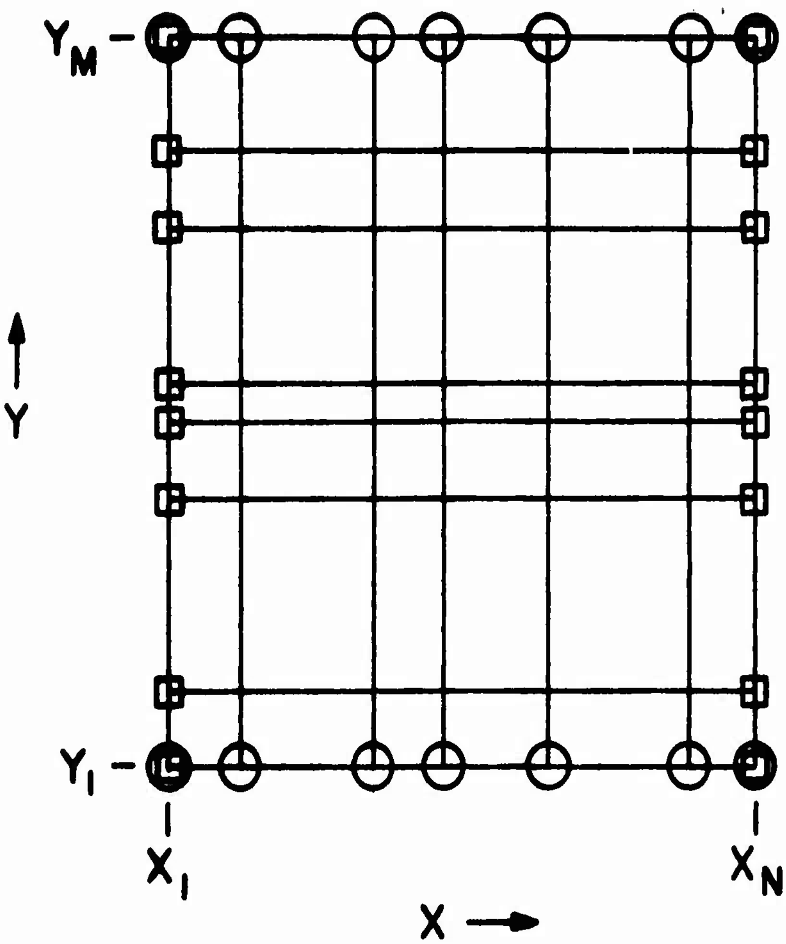

Figure 1

(2) The second problem is, assuming the normal derivatives along the boundaries as discussed above have been given or estimated, to fit a "smooth function" $u(x,y) \in C^2$ (twice continuously differentiable) through these values.

The bicubic spline interpolation routines BICUB1, BICUB2, GETBP, AND SOLVIT described later implement a bicubic spline interpolation technique[2] [1] (see Section 3.15) which yields a piecewise bicubic polynomial $u(x,y)$. This function is defined in each rectangular cell,

$$R_{ij} : x_{i-1} \leq x \leq x_i \; ; \; y_{i-1} \leq y \leq y_i \; , \qquad (1)$$

of the grid as

$$u(x,y) = C_{ij}(x,y) = \sum_{m,n=0}^{3} \gamma_{mn}^{ij} (x-x_{i-1})^m \, (y-y_{i-1})^n \qquad (2)$$

where $(x,y) \in R_{ij}$.

<u>Individual Subroutine Functions</u>

<u>BICUB1</u> - A subroutine for calculating the normal derivatives at each mesh point.

<u>BICUB2</u> - A subroutine for interpolating a value, $u(x,y)$, at any point (x,y) within the region subtended by the mesh.

<u>EDGES</u> - A subroutine for estimating the required normal derivatives along the boundaries assuming they have not been given, using a moving third order two dimensional Lagrange interpolating polynomial.

---

2. Note that Eq.(10) of [1] contains a typographical error and should read

$$A(\Delta x_{i-1}) K_{ij} \, A^T (\Delta y_{j-1}) = \Gamma_{ij},$$ where T indicates a transpose. Also, the 8th character on line 4, p212 of [1] should be I and not 1.

GETBP - A subroutine for calculating the two dimensional difference arrays b and b' of Eq.(15) of [1].

LAGRAN - A subroutine for determining the value of a two dimensional Lagrange interpolating polynomial of degree m in x and n in y and its derivatives with respect to x, with respect to y, and with respect to both x and y at any intersection of a two dimensional mesh defined by (m+1) levels of x_i, i = a, ..., a+m and (n+1) levels of y_j, j = c, ..., c+n. See Eq.(3).

SOLVIT - A subroutine for solving a linear system using Gaussian elimination as illustrated in Eqs.(15) and (16) of [1].

2.2 Problem Background

In the development of models of certain physical phenomena it is frequently useful to obtain a smooth functional representation of a quantity which is known at only a discrete set of points over a two-dimensional domain. Frequently, it is required that this function have continuous first and second derivatives. Once such a representation is obtained, it is possible to differentiate and integrate it in closed form.

A major problem with a Lagrange interpolating polynomial defined over a N x M mesh is that it must maintain (N-2) in x and (M-2) in y continuous derivatives and still pass through all of the data points. These requirements can and generally do lead to many large and unrealistic mountains and valleys in the interpolation surface, i.e., large interpolation errors. This problem is significantly reduced, but not completely eliminated by the bicubic spline (See Section 3.13).

A secondary, but still significant, problem is that when a large number of points is used, the evaluation and calculation of the Lagrange interpolating polynomial is costly and unreliable ([6] , p. 231). For further discussion see [8].

3.0 USAGE

3.1 Calling Sequence or Operational Procedure

```
BICUB1(N,M,NDFAULT,X,Y,U,P,Q,S,MAX,W1,W2,W3,W4
       W5,W6,W7)

BICUB2(XPT,YPT,UPT,NERROR,N,M,X,Y,U,P,Q,S)
```

3.2 Arguments, Parameters, and/or Initial Conditions

N -- number of x points at which the function was observed.

($N \geq 4$). TYPE INTEGER.

M -- number of y points at which the function was observed.

($M \geq 4$). TYPE INTEGER.

NDFAULT -- a parameter that must be set to 1 if subroutine BICUB1 is to call subroutine EDGES to calculate the "required" normal derivatives along the boundaries of the mesh. If NDFAULT is not set to 1, subroutine BICUB1 assumes the normal derivatives for the boundaries have already been entered into arrays, P,Q, and S, by the user's calling program. TYPE INTEGER.

The "required" normal derivatives are:
P(I,J) for I=1 and N; J=1 to M.
Q(J,I) for J=1 and M; I=1 to N.
S(J,I) for I=1 and N; J=1 and M.

X -- the vector of distinct values of the first independent variable arranged in ascending order; (x_i for i =1 to N). The minimum length of X is 4 and the maximum length is determined by the amount of core available. DIMENSION X(N). TYPE REAL.

Y -- the vector of distinct values of the second independent variable arranged in ascending order. (y_j for j=1 to M). The minimum length of Y is 4 and the maximum length is determined by the amount of core available. DIMENSION Y(M). TYPE REAL.

U -- the matrix of function values corresponding to X and Y, i.e., U(I,J) is u_{ij} of Section 2.1. DIMENSION U(N,M). TYPE REAL.

P -- the matrix of normal derivatives with respect to x corresponding to X and Y; i.e., P(I,J) is $U_x(X_i,Y_j)$. If NDFAULT is not 1, P(I,J) for (I=1 and N; J=1 to M) are required input. If NDFAULT is set to 1, no values of P are required as they will be calculated by subroutine EDGES. DIMENSION P(N,M). TYPE REAL.

Q -- the matrix of normal derivatives with respect to y corresponding to X and Y; i.e., Q(J,I) is $U_y(X_i, Y_j)$. Note the inversion of the J and I indices in Q. If NDFAULT is not 1, Q(J,I) for (J=1 and M;I=1 to N) are required input. If NDFAULT is set to 1, no values of Q are required as they will be calculated by subroutine EDGES. DIMENSION Q(M,N). TYPE REAL.

S -- the matrix of normal derivatives with respect to both x and y corresponding to X and Y: i.e., S(J,I) is $U_{xy}(X_i, Y_j)$. Note the inversion of indices I and J in S. If NDFAULT is not 1, S(J,I) for (I=1 and N; J=1 and M) are required input. If NDFAULT is set to 1, no values of S are required as they will be calculated by subroutine EDGES. DIMENSION S(M,N). TYPE REAL.

MAX -- the greater of N and M. TYPE INTEGER.

W1,...,W7 -- seven arrays which are used as working areas by subroutine BICUB1. Each of these arrays must be dimensioned to MAX words in the user's program. The user does not assign values to these arrays. TYPE REAL.

XPT -- the X coordinate of the point to be interpolated. TYPE REAL.

YPT -- the Y coordinate of the point to be interpolated. TYPE REAL.

UPT -- the interpolated value to be obtained. TYPE REAL.

NERROR -- an error indicator. If the point (XPT,YPT) does not lie within the mesh, NERROR will be set to 1 by BICUB2; otherwise it will remain set to 0. IF NERROR is returned as 1, the interpolated value is set to -0.0 and an error message is printed. See Section 3.5. TYPE INTEGER.

3.3 Space Required (Decimal and Octal)

3.3.1. Unique Storage (exclusive of system library)

Subroutine		Decimal	Octal
BICUB1		1083	2073
BICUB2		570	1072
EDGES		474	732
GETBP		183	267
LAGRAN		588	1114
SOLVIT		129	201
	TOTAL	3027	5723

To interpolate a function over an N X M mesh the user's program is required to dimension the arrays X,Y,U,P,Q,S, and W1 through W7. These arrays require a total of

$$4NM + N + M + 7K \text{ words,}$$

where K is the greater of N and M.

3.3.2. Common Blocks

None.

3.3.3. Temporary Storage

Once subroutine BICUB1 has been called and the elements of the U,P,Q,S arrays have been determined, the space occupied by the working arrays W1,W2,...,W7 can be used for other purposes. That is, the 7K term of the above equation may be deleted.

3.4 Messages and Instructions to the Operator

None.

3.5 Error Returns, Messages, and Codes

Subroutine BICUB1 may print out the following error messages:

```
"ERROR - THE Y VECTOR IS NOT ARRANGED PROPERLY.
         ERROR DETECTED BY BICUB1.
         THE Y VECTOR IS (listed)"

"ERROR - THE X VECTOR IS NOT ARRANGED PROPERLY.
         ERROR DETECTED BY BICUB1.
         THE X VECTOR IS (listed)."

"ERROR - THE PARAMETER MAX OF SUB. BICUB1 WAS SET
         TO __.  IT SHOULD BE __."

"ERROR - THE X VECTOR HAS __POINTS AND THE MINIMUM
         ALLOWED IS 4."

"ERROR - THE Y VECTOR HAS __POINTS AND THE MINIMUM
         ALLOWED IS 4."
```

Subroutine BICUB2 will print out one or more of the following messages if an attempt is made to interpolate a point beyond the boundaries of the mesh:

```
"ERROR - XPT OUT OF BOUNDS
         DETECTED BY SUB. BICUB2"

"ERROR - YPT OUT OF BOUNDS
         DETECTED BY BICUB2"
```

3.6 Informative Messages to the User

None.

3.7 Input

No data are input. See Section 3.2.

3.8 Output

(1) Completion of the P, Q and S arrays.

(2) The value of the function $u(x,y)$ for any given (x,y) within the domain.

3.9 Formats

Not applicable.

3.10 External Routines and Symbols

BICUB1	-	EDGES	(deck)
		GETBP	"
		SOLVIT	"
EDGES	-	LAGRAN	"

3.11 Timing

The time required by subroutine BICUB1 is dependent upon the mesh size and whether or not the normal derivatives along the boundary are known. In the example of Section 7.0 BICUB1 took approximately 23 milliseconds for a 5 x 6 mesh when the boundary derivatives were known and approximately 135 milliseconds when they were unknown.

The time required for a call to BICUB2 is dependent on the mesh size. In the example of Section 7.0 an average call took about 3 milliseconds.

These time estimates should be considered very rough because of the method used to obtain them and the inaccuracies of the timing function (TIMELEFT) used.

3.12 Accuracy

An excellent discussion of the errors involved in bicubic spline interpolation is given in the paper by G. Birkhoff and C. de Boor [8]. Here, we will simply mention that for the special case of a 4 x 4 mesh, the interpolated values u(x,y) will agree exactly, as they must, with those obtained from a third order two dimensional Lagrange interpolating polynomial. In this case, the remainder term is well known [3]. The final accuracy is dependent upon both discretion and rounding errors. A rough order of magnitude for these errors may be obtained from Section 7.0. The reader is referred to [4 and 5] for further discussion.

3.13 Cautions to Users

If the values $[u_{ij}]$ are highly variable along i or j, the interpolation surface may be forced to have unusually high mountains and deep valleys in order to maintain two continuous derivatives and still pass through all the data points. In fact, some interpolated values of u may be so large or so small as to be physically unrealistic. Whether or not this is the case will depend on the particular problem. The author has found that plotting several interpolated values, between and together with the given values, along a fixed direction in the x-y plane is helpful in detecting such conditions. In any event, care should be taken as the interpolation process could cause a physical model to generate faulty predictions.

3.14 Program Deck Structure

$^7_9$ JOB card

$^7_9$ FTN card

Users program (containing calls to BICUB1 and BICUB2)

Subroutine BICUB1
Subroutine BICUB2
Subroutine EDGES
Subroutine GETBP
Subroutine LAGRAN
Subroutine SOLVIT

} E_1 - NRL - BICUBIC

SCOPE card

$^7_9$ LOAD card

$^7_9$ RUN card .

EØF

3.15 References

[1]. C.de Boor, "Bicubic Spline Interpolation", J. of Mathematics and Physics, 41, 212-218 (1962).

[2]. B. Carnahan, H. Luther, and J. Wilkes, Applied Numerical Methods, (Wiley and Sons, New York, 1969), Chapter 1.

[3]. B. Carnahan, et. al., p65, problems 1.35 and 1.38.

[4]. J. H. Wilkinson, Rounding Errors in Algebraic Processes, (Prentice Hall, N. J., 1968).

[5]. J. M. Ortega, Numerical Analysis (A Second Course). (Academic Press, New York, 1972) Chapters 1, 7 and 9.

[6]. C.de Boor and S. D. Conte, Elementary Numerical Analysis: An Algorithmic Approach, 2nd ed., (McGraw Hill, New York, 1972). Pages 231-240 describe one dimensional cubic spline interpolation.

[7]. C. Price, "Table Lookup Techniques", Computing Surveys, 3, No. 2 (June 1971) pp53-56.

[8]. G. Birkhoff and C. R. de Boor, "Piecewise Polynomial Interpolation and Approximation", in Approximation of Functions, H. Garabedian (editor), (Elsevier Publishing Co., Amsterdam, 1965).

[9]. C. de Boor, Private communication.

[10]. H. Späth, "Algorithm 10, Two Dimensional Smooth Interpolation", Computing 4, 178-182 (1962). (In German).

[11]. H. Späth, "Correction to Algorithm 10", Computing 8, 200-201 (1971). (In German).

4.0 METHOD OR ALGORITHM

4.1 Subroutine BICUB1

We begin by considering the problem of estimating the required boundary derivatives (see Section 3.2) under the assumption that they are unknown.

Consider a 3rd order two dimensional Lagrange interpolating polynomial over a moving (a and c are variable) 4 x 4 submesh, i.e.,

$$v(x,y) = \sum_{i=a}^{b} \sum_{j=c}^{d} X_i(x) Y_j(y) u_{ij} \tag{3}$$

where $b = a + 3$, $d = c + 3$,

u_{ij} is defined in Section 2.1 ,

$$X_i(x) = \prod_{\substack{k=a \\ k \neq i}}^{b} \frac{x - x_k}{x_i - x_k} ,$$

$$Y_j(y) = \prod_{\substack{k=c \\ k \neq j}}^{d} \frac{y - y_k}{y_j - y_k} .$$

Differentiating Eq. (3) we can obtain closed form expressions for $v_x(x,y)$, $v_y(x,y)$, and $v_{xy}(x,y)$.

Subroutine LAGRAN can evaluate these expressions at any mesh point.

Basically, subroutine EDGES moves the 4 x 4 submesh of Eq. (3) along the boundaries of Figure 1 while calling subroutine LAGRAN to obtain the required normal derivatives.

Once the required boundary derivatives are obtained, the rest of the derivatives, p_{ij}, q_{ij}, and S_{ij}, are obtained for each ij mesh point by using the algorithm described in [1] , pages 217-218.

4.2 Subroutine BICUB2

To interpolate a value, $u(x,y)$, at the point (x,y), subroutine BICUB2 begins by performing a binary search to determine the ij rectangle in which the point lies. A binary search [7] is used on the assumption that for most problems the X and Y vectors will be large enough to exceed the break even point between sequential and binary searches (about 50 points).

Once i and j are determined, a cubic Hermite basis is used to evaluate $u(x,y)$. That is,

$$u(x,y) = \sum_{r,s=1}^{4} Q_{rs}^{ij} \phi_r(x,h) \psi_s(y,k) \tag{4}$$

where

$$Q^{ij} = \begin{bmatrix} u_{i-1,j-1} & u_{i-1,j} & q_{i-1,j-1} & q_{i-1,j} \\ u_{i,j-1} & u_{i,j} & q_{i,j-1} & q_{i,j} \\ p_{i-1,j-1} & p_{i-1,j} & s_{i-1,j-1} & s_{i-1,j} \\ p_{i,j-1} & p_{i,j} & s_{i,j-1} & s_{i,j} \end{bmatrix} \tag{5}$$

$$h = x_i - x_{i-1} \tag{6}$$

$$x' = x - x_{i-1} \tag{7}$$

$$k = y_j - y_{j-1} \tag{8}$$

$$y' = y - y_{j-1} \tag{9}$$

$$\phi_1(x,h) = 1 + \left(\frac{x'}{h}\right)^2\left(\frac{2x'}{h} - 3\right) \tag{10}$$

$$\phi_2(x,h) = \left(\frac{x'}{h}\right)\left(3 - \frac{2x'}{h}\right) \tag{11}$$

$$= 1 - \phi_1(x,h)$$

$$\phi_3(x,h) = \left(\frac{x'}{h}\right)\frac{(h-x')^2}{h} \tag{12}$$

$$\phi_4(x,h) = \left(\frac{x'}{h}\right)^2 (x' - h) \tag{13}$$

$$= -\phi_3\ (h-x)$$

The functions ψ_s, for $s = 1$ to 4, are obtained by replacing ϕ, x', and h by ψ, y', and k respectively in Eqs. (10) through (13).

This procedure requires the storage of four values (u_{ij}, p_{ij}, q_{ij}, and S_{ij}) for each mesh point. And the evaluation of $u(x,y)$ by BICUB2 requires 32 additions/subtractions and 27 multiplications/divisions. Assuming a multiplication/division is equivalent in time to three additions/subtractions, this results in 113 "operations".

An alternative approach, not taken in this report, would be to convert to a local power basis. In particular, calculate the 16 values of γ_{mn}^{ij} (see Eq.(2)) for each ij rectangle, as described in [1], and store them for each ij rectangle of the mesh. This would require approximately four times as much storage as the above method. Also about 52 additions/subtractions and 76 multiplications/divisions (270 "operations") would be required to obtain the 16 coefficients γ_{mn}^{ij}, for $m,n = 0$ through 3.

The advantage of this approach is that only about 19 additions/subtractions and 15 multiplications/divisions (64 "operations") would be required by BICUB2 for the evaluation. This suggests that, for very fine mesh evaluations, in which every bicubic polynomial is evaluated on the average six or more times, it is more efficient to obtain the local power basis coefficients, γ_{mn}^{ij}, for the entire mesh once and save them. Of course, this results in a rather severe penalty in storage.

In contrast, by using the Hermite basis, as we've done here, evaluation costs slightly more work but considerably less storage.

In summary, it seems best for most applications to save only the partials at the mesh points and use the Hermite basis approach.

5.0 SOURCE LANGUAGE LISTING

```
C       IDENT NUMBER E1001000
C       TITLE - BICUBIC SPLINE INTERPOLATION
C       IDENT NAME - E1-NRL-BICUBIC
C       LANGUAGE -FORTRAN
C       COMPUTER - CDC-3800
C
C       CONTRIBUTOR - JOHN J CORNYN
C                     CODE 5493
C                     INFORMATION PROCESSING SYSTEMS BRANCH
C                     COMMUNICATIONS SCIENCES DIVISION
C       ORGANIZATION -NRL - NAVAL RESEARCH LABORATORY
C                           WASHINGTON, D.C. 20375
C
C       DATE - FEBRUARY 1973
C
C ******************************************************************
C       PURPOSE
C
C       SUBROUTINES BICUB1 AND BICUB2 PERFORM BICUBIC SPLINE
C       INTERPOLATION OF A TABULATED FUNCTION OF TWO VARIABLES.
C
C ******************************************************************
C
C     REFERENCE - DE BOOR, C. - BICUBIC SPLINE INTERPOLATION
C                               J. OF MATHEMATICS AND PHYSICS,
C                               VOL 41,PP 212-218,(1962)
C
C ******************************************************************
C
C  SUBROUTINE CALLING ORDER ---
C
C     THE USER-S PROGRAM CALLS SUBROUTINES BICUB1 AND/OR BICUB2.
C     SUBROUTINE BICUB1 CALLS SUBROUTINES  EDGES,GETBP AND SOLVIT.
C     SUBROUTINE EDGES  CALLS SUBROUTINE LAGRAN.
C     SUBROUTINES BICUB2,GETBP,SOLVIT, AND LAGRAN DO NOT CALL
C              ANY OTHER SUBROUTINES.
C ******************************************************************
C
C******************************************************************
C
C  DICTIONARY FOR BICUBIC SPLINE INTERPOLATION
C ***********
C
C    DICTIONARY CODE - THE SECOND LETTER OF A LINE OF THE DICTIONARY
C                      INDICATES THE TYPE OF ENTRY BEING DESCRIBED.
C                      THIS CODE IS AS FOLLOWS
C          DICTIONARY CODE      TYPE OF VARIABLE
C               CA               ABREVIATION
C               CI               INTEGER VARIABLE
C               CIA              INTEGER VARIABLE ARRAY
C               CR               REAL VARIABLE
C               CRA              REAL VARIABLE ARRAY
C               CS               ROUTINE
C               CT               WORD FROM TEXT
C
C  FOLLOWING THE DICTIONARY CODE IS A NUMBER WHICH INDICATES THE
```

```
C    SUBROUTINE NUMBER(S) IN WHICH THE ENTRY APPEARS
C          SUBROUTINE NAME     SUBROUTINE NO.
C               BICUB1              1
C               BICUB2              2
C               EDGES               3
C               GETBP               4
C               LAGRAN              6
C               SOLVIT              7
C               CHECK               8
C
C     FOR EXAMPLE, CR2,3 MEANS  THE ENTRY  APPEARS IN SUBROUTINES
C     BICUB2 AND EDGES AND IS A REAL VARIABLE.
C
C
C ****************************************************************
CS1,8          BICUB1   = A SUBROUTINE FOR CALCULATING THE NORMAL
C                         DERIVATIVES AT EACH MESH POINT.
CS2,8          BICUB2   = A SUBROUTINE FOR INTERPOLATING A VALUE
C                         UPT AT ANY POINT (XPT,YPT) WITHIN OR ON
C                         THE MESH.
CRA1,4,7       BIM1(I)  = HOLDS THE ELEMENT B(I,I-1) OF EQ.15 OF REF 1
CRA1,4,7       BIP1(I)  = HOLDS THE ELEMENT B(I,I+1) OF EQ.15 OF REF 1
CRA1,4,7       BP(I)    = HOLDS THE ELEMENT B(I,I) OF EQ. 15 OF REF 1.
C                         THE ARRAYS BIM1,BIP1, AND BP ARE USED TO HOLD
C                         THE B AND B PRIME DIFFERENCE ARRAYS GIVEN
C                         IN THE REFERENCE. THESE ARRAYS ARE USED FOR
C                         GAUSSIAN ELIMINATION.
CS8            CHECK    = A SAMPLE PROGRAM  TO CHECK OUT THE
C                         BICUBIC SPLINE PACKAGE
CR2            CX1      = X DEPENDENT TEMPORARY VARIABLE
CR2            CX2      = AN X DEPENDENT TEMPORARY VARIABLE
CR2            CX3      = AN X DEPENDENT TEMPORARY VARIABLE
CR2            CY1      = A  Y DEPENDENT TEMPORARY VARIABLE
CR2            CY2      = A  Y DEPENDENT TEMPORARY VARIABLE
CR2            CY3      = A  Y DEPENDENT TEMPORARY VARIABLE
CRA1,7         D(I)     = CORRESPONDS TO THE D MATRIX OF EQ. 13 OF THE
C                         REFERENCE
CR6            DENOM    = A DENOMINATOR
CR8            DIFF     = DIFFENCE BETWEEN INTERPOLATED AND EXACT VALUES
CRA1,7         DP(I)    = CORRESPONDS TO THE D PRIME  VECTOR OF EQ.16
C                         OF THE REF.
CR6            DPT      = THE ANSWER GIVEN BY LAGRAN
CRA1           DS(I)    = AN ARRAY FOR HOLDING DIFFERENCES IN SOLVING
C                         EQ 12 AND 14 OF REF.(DEBOOR)
CR4            DXL      = DELTA X LEFT
CR4            DXR      = DELTA X RIGHT
CS1,3          EDGES    = A SUBROUTINE FOR ESTIMATING THE REQUIRED
C                         NORMAL DERIVATIVES ALONG THE MESH BOUNDARIES
C                         ASSUMING THEY HAVE NOT BEEN GIVEN, USING
C                         A  THIRD ORDER TWO DIMENSIONAL LAGRANGE
C                         INTERPOLATING POLYNOMIAL
CS1,4          GETBP    = A SUBROUTINE FOR CALCULATING THE TWO
C                         DIMENSIONAL DIFFERENCE ARRAYS B AND B PRIME
C                         OF EQ 15 OF REF.(DEBOOR)
CI1-8          I        = INDEX,   GENERALLY USED FOR THE X ARRAY
CI1,4,2,7      IM1      = I MINUS ONE
```

```
CI1,4,7      IP1     = I PLUS ONE
CI4          IP2     = I PLUS 2
CI1-3,6-7,8  J       = INDEX GENERALLY USED FOR THE Y ARRAY
CI1,2        JM1     = J MINUS ONE
CI1          JP1     = J PLUS ONE
CI6,8        K       = AN INDEX
CI2          KH      = UPPER LIMIT FOR BINARY SEARCH
CI2          KL      = LOWER LIMIT FOR BINARY SEARCH
CI1,6        L       = A COUNTER
CI1          L1      = L PLUS ONE
CS3,6        LAGRAN  = A SUBROUTINE FOR DETERMINING THE VALUE OF A
C                      TWO DIMENSIONAL LAGRANGE INTERPOLATING
C                      POLYNOMIAL OF ARBITRARY DEGREE IN X AND Y,
C                      AND ITS DERIVATIVES WITH RESPECT TO X,
C                      WITH RESPECT TO Y,AND WITH RESPECT TO BOTH
C                      X AND Y AT ANY INTERSECTION POINT OF A
C                      X ANDY AT ANY INTERSECTION POINT OF A
C                      TWO DIMENSIONAL MESH,
CI1-3,  6,8  M       = THE NUMBER OF Y POINTS AT WHICH THE FUNCTION
C                      WAS OBSERVED, MUST BE GREATER THAN 3,
CI1,8        MAX     = THE GREATER OF N AND M,
C
CI1          MAXS    = THE VALUE MAX SHOULD HAVE BEEN SET TO,
CI3,6        MF      = NUMBER OF THE FINAL POINT ON Y AXIS TO
C                      BE USED IN LAGRANGE INTERPOLATION
CI1,8        MM1     = M MINUS ONE
CI1          MM2     = M MINUS TWO
CI1          MMIN    = THE SMALLEST VALUE M CAN TAKE,
CI6          MPT     = THE VALUE OF THE Y VECTOR TO BE USED IN LAGRANGE
C                      INTERPOLATION
CI3,6        MS      = NUMBER OF THE STARTING POINT ON Y AXIS TO
C                      BE USED IN LAGRANGE INTERPOLATION
CI1-3,  6,8  N       = THE NUMBER OF X POINTS AT WHICH THE FUNCTION
C                      WAS OBSERVED,
CI7,4        N       = NUMBER OF ELEMENTS IN LINEAR SYSTEM
CI1,8        NDFAULT = A PARAMETER WHICH MUST BE SET TO 1 IF SUBROUTINE
C                      BICUB1 IS TO CALL EDGES TO CALCULATE THE
C                      REQUIRED NORMAL DERIVATIVES ALONG THE BOUNDARIES
C                      OF THE MESH, IF NDFAULT IS NOT SET TO 1, BICUB1
C                      ASSUMES THE NORMAL DERIVATIVES FOR THE
C                      BOUNDARIES HAVE ALREADY BEEN ENTERED INTO
C                      ARRAYS P,Q, AND S BY THE USERS CALLING PROGRAM,
CI2,8        NERROR  = AN ERROR INDICATOR, IF THE POINT (XPT,YPT)
C                      DOES NOT LIE WITHIN THE MESH, NERROR WILL
C                      BE SET TO 1 BY BICUB2  OTHERWISE IT WILL
C                      REMAIN SET TO 0, IF NERROR IS RETURNED AS 1,
C                      AN INTERPOLATED VALUE IS NOT COMPUTED,
CI3,6        NF      = NUMBER OF THE FINAL POINT ON THE X AXIS TO
C                      BE USED IN LAGRANGE INTERPOLATION,
CI1,4,7,8    NM1     = N MINUS ONE
CI1,4        NM2     = N MINUS TWO
CI4          NM3     = N MINUS 3
CI1          NMIN    = THE SMALLEST VALUE N CAN TAKE
CI6          NPT     = THE VALUE OF THE X VECTOR TO BE USED IN LAGRANGE
C                      INTERPOLATION, XPT=X(NPT)
CI3,6        NS      = NUMBER OF THE STARTING POINT ON THE X AXIS TO
```

```
C                          BE USED IN LAGRANGE INTERPOLATION.
CI6             NTYPE    = SET TO 1 IF LAGRAN IS TO INTERPOLATE A
C                          VALUE OF THE FUNCTION AT THE (NPT,MPT)MESH
C                          POINT.
C                          SET TO 2  TO GET PARTIAL DERIVATIVE W/R TO X.
C                          SET TO 3  TO GET  PARTIAL DERIVATIVE W/R TO Y.
C                          SET TO 4  TO GET PARTIAL DERIVATIVE W/R TO
C                          BOTH X AND Y.
CRA1-3,   6,8   P        = THE  NORMAL  DERIVATIVES OF U WITH RESPECT TO X
C
CR2             PI1J1    = P(I-1,J-1)
CR2             PIJ1     = P(I,J-1)
CRA1-3,   6,8   Q        = THE  NORMAL  DERIVATIVES OF U WITH RESPECT TO Y.
CR1             R        = A TEMPORARY VARIABLE USED IN FORMING
C                          THE C MATRIX
CS8             RANF     = A CDC 3800 RANDOM NUMBER GENERATOR
C                          GENERATES UNIFORMLY DISTRIBUTED RANDOM
C                          NUMBERS BETWEEN 0 AND1
CR1             RINV     = THE INVERSE OF R
CRA1-3,   6,8   S        = THE  NORMAL  DERIVATIVES OF U WITH RESPECT TO
C                          BOTH X  AND Y.
CS7,1           SOLVIT   = A SUBROUTINE FOR SOLVING A LINEAR SYSTEM
C                          USING GAUSSIAN ELIMINATION AS ILLUSTRATED
C                          IN SC, 15 AND 16 OF REF (DEBOOR).
CRA1            STEMP    = A TEMPORARY ARRAY USED IN SOLVING FOR S
CR6             SUM      = A SUMMRTION
CR6             SUMX     = A SUMMATION ALONG X
CR6             SUMY     = A SUMMATION ALONG Y
CR8             T        = VARIABLE FOR TIME CALCULATION
CR2             TI       = TEMPORARY VARIABLE IN EVALUATION
CR2             TIM1     = TEMPORARY VARIABLE IN EVALUATION
CS8             TIMELEFT = A CDC-388 SYSTEM LIBRARY FUNCTION
C                          GIVING THE NUMBER OF SECONDS LEFT UNTIL THE
C                          THE TIME LIMIT FOR THE JOB WILL BE REACHED.
CR2             TPI      = TEMPORARY VARIABLE IN EVALUATION
CR2             TPIM1    = TEMPORARY VARIABLE IN EVALUATION
CRA1-3,   6,8   U        = THE ARRAY OF FUNCTION VALUES CORRESPONDING
C                          TO X AND Y.
CR8             UEXACT   = VALUE OF U OBTAINED BY EVALUATING THE
C                          ARBITRARY POLYNOMIAL AT THE COORDINTES (XI,YI)
CR2             UI1J1    = U(I-1,J-1)
CR2             UIJ1     = U(I,J-1)
CR8             UINT     = VALUE OF U OBTAINED BY BICUBIC SPLINE
C                          INTERPOLATION AT THE COORDINATES (XI,YI)
CR2             UPT      = THE INTERPOLATED VALUE AT  (XPT,YPT)
CRA4            V        =  A VECTOR OF VALUES
CR8             W1       = WORKING  ARRAYS  DIMENSIONED TO
CR8             W2         MAX WORDS IN THE USERS PROGRAM.
CR8             W3
CR8             .
CR8             .
CR8             W7
CRA1-3,   6,8   X        = THE VECTOR OF DISTINCT VALUES OF THE FIRST
C                          INDEPENDENT VARIABLE ARRANGED IN ASCENDING
C                          ORDER,THE MINIMUM LENGTH OF X IS 4 AND THE
C                          MAXIMUM LENGTH IS DETERMINED BY THE AMOUNT
```

```
C                              OF CORE AVAILABLE.
CR1             X32       = X(3)MINUS X(2)
CR8,6           XI        = THE ITH VALUE OF X
CR2             XIM1      = X(I-1)
CR2             XMX1OM    = TEMPORARY VARIABLE
CR1             XN12      = X(NM1)-X(NM2)
CR6             XP        = X(NPT)
CR2             XPT       = THE X CO-ORDINATE OF THE POINT TO BE
C                           INTERPOLATED.
CRA1-3,   6,8   Y         = THE VECTOR OF DISTINCT VALUES OF THE SECOND
C                           INDEPENDENT VARIABLE ARRANGED IN ASCENDING
C                           ORDER,THE MINIMUM LENGTH OF Y IS 4
C                           AND THE MAXIMUM LENGTH IS DETERMINED BY
C                           THE AMOUNT OF CORE AVAILABLE.
CR1             Y32       = Y(3)-Y(2)
CR8             YI        = THE ITH VALUE OF Y
CR6             YJ        = Y(J)
CR2             YJM1      = Y(J-1)
CR1             YM12      = Y(MM1)-Y(MM2)
CR2             YMY1OK    = TEMPORARY VARIABLE
CR6             YP        = Y(MPT)
CR2             YPT       = THE Y CO-ORDINATE OF THE POINT TO BE
C                           INTERPOLATED.
CRA7            Z         = THE SOLUTION VECTOR FOR THE LINEAR SYSTEM
C
C ******************************************************************
C*******************************************************************
C
      SUBROUTINE BICUB1(N,M,NDFAULT,X,Y,U,P,Q,S,MAX,BP,BIP1,BIM1,D,DS,
     X STEMP,DP)
C
C BICUBIC SPLINE INTERPOLATION
C THIS SUBROUTINE CALCULATES THE PARTIAL DERIVATIVES FOR THE MESH
C
      DIMENSION X(N),Y(M),U(N,M),P(N,M),Q(M,N),S(M,N)
      DIMENSION BP(MAX),BIP1(MAX),BIM1(MAX),D(MAX),DS(MAX),STEMP(MAX),
     X DP(MAX)
C
      DATA(NMIN=4),(MMIN=4)
C
C CHECK TO SEE IF THE MAX PARAMETER WAS SET CORRECTLY
C
C
C MAXS IS WHAT MAX SHOULD BE
C
      MAXS=N
      IF(N.LT.M) MAXS=M
      IF(MAXS .NE. MAX) GO TO 900
C
C
C DETERMINE WHETHER THE X AND Y VECTORS ARE WITHIN LIMITS
C
C
      IF(N .LT. NMIN) GO TO 905
      IF(M .LT. MMIN) GO TO 907
      NM1=N-1
```

```
      NM2=N-2
      MM1=M-1
      MM2=M-2
C
C DETERMINE WHETHER THE X AND Y ARRAYS CONTAIN DISTINCT ELEMENTS AND
C ARE ARRANGED IN ASCENDING ORDER.
C
      DO 200 I=1,NM1
      IF(X(I) .GE. X(I+1)) GO TO 911
  200 CONTINUE
C
      DO 210 J=1,MM1
      IF(Y(J) .GE. Y(J+1)) GO TO 912
  210 CONTINUE
C
C DETERMINE THE EDGE BOUNDARIES FOR P,Q,AND S IF REQUESTED
C
      IF(NDFAULT .EQ. 1) CALL EDGES(N,M,X,Y,U,P,Q,S)
C
C GET THE DIFFERENCE ARRAY,B, FOR THE X VECTOR AND ALSO
C GET THE B PRIME ARRAY,BP, FROM THE B ARRAY
C
      CALL GETBP(N,X,BP,BIP1,BIM1)
C
C NOW SOLVE FOR THE PARTIALS W/R TO X WHICH WERE NOT GIVEN
C
      X32=X(3)-X(2)
      XN12=X(NM1)-X(NM2)
      DO 30 J=1,M
C
C SET UP THE D VECTOR FOR EQ(11) OF REF
C
      DO 35 IM1=1,NM2
      I=IM1+1
      IP1=I+1
      R=(X(I)-X(IM1))/(X(IP1)-X(I))
      RINV=1./R
      D(IM1)=3.*(R*(U(IP1,J)-U(I,J))+RINV*(U(I,J)-U(IM1,J)))
      IF(J .EQ. 1 .OR. J.EQ. M) 34,35
C
   34 DS(IM1)=3.*(R*(Q(J,IP1)-Q(J,I))+RINV*(Q(J,I)-Q(J,IM1)))
C
C NOTE Q AND S ARRAYS ARE STORED AS Q(J,I) AND S(J,I) RATHER THAN
C Q(I,J) BECAUSE OF FORTRAN CONVENTIONS FOR STORING ARRAYS COLUMNWISE
C
   35 CONTINUE
C
C ADD ADDITIONAL TERMS TO THE FIRST AND LAST ELEMENTS OF THE D VECTOR
C
      D(1)=D(1)-X32*P(1,J)
      D(NM2)=D(NM2)-XN12*P(N,J)
C
C
C NOW SOLVE LINEAR SYSTEMS FOR EQ(11) OF REFERENCE.
C
      CALL SOLVIT(NM2,P(2,J),D,BP,BIP1,BIM1,DP)
```

```
C
      IF(J .EQ. 1 .OR. J .EQ. M) 37,30
C
C ADD ADDITIONAL TERMS TO THE DS ARRAY
C
   37 DS(1)=DS(1)-X32*S(J,1)
      DS(NM2)=DS(NM2)-XN12*S(J,N)
C
C NOW SOLVE LINEAR SYSTEM FOR EQ(12)
C
      CALL SOLVIT(NM2,STEMP,DS,BP,BIP1,BIM1,DP)
C
C MOVE VALUES FROM THE TEMPORARY ARRAY INTO  THE PARTIAL
C ARRAY FOR THE CROSS TERMS, WE DO THIS BECAUSE OF FORTRAN ARRAY
C STORAGE CONVENTIONS
C
      DO 10 L=1,NM2
      L1=L+1
      S(J,L1)=STEMP(L)
   10 CONTINUE
C
   30 CONTINUE
C
C GET THE DIFFERENCE ARRAY ,B, FOR THE Y VECTOR AND THE B PRIME ARRAY
C
      CALL GETBP(M,Y,BP,BIP1,BIM1)
C
C NOW GET THE PARTIALS WITH RESPECT TO Y
C WHICH WERE NOT GIVEN, I. E. MEMBERS OF THE G ARRAY WHICH WERE NOT
C SPECIFIED
C
      Y32=Y(3)-Y(2)
      YM12=Y(MM1)-Y(MM2)
      DO 40 I=1,N
C
C SET UP THE D MATRIX FOR EQ(13) OF REF
C
      DO 45 JM1=1,MM2
      J=JM1+1
      JP1=J+1
      R=(Y(J)-Y(JM1))/(Y(JP1)-Y(J))
      RINV=1./R
      D(JM1)=3.*(R*(U(I,JP1)-U(I,J))+RINV*(U(I,J)-U(I,JM1)))
      DS(JM1)=3.*(R*(P(I,JP1)-P(I,J))+RINV*(P(I,J)-P(I,JM1)))
   45 CONTINUE
C
      D(1)=D(1)-Y32*G(1,I)
      D(MM2)=D(MM2)-YM12*G(M,I)
C
      DS(1)=DS(1)-Y32*S(1,I)
      DS(MM2)=DS(MM2)-YM12*S(M,I)
C
C NOW CAN SOLVE THE LINEAR SYSTEM OF EQ(13) FOR THIS I
C
      CALL SOLVIT(MM2,G(2,I),D,BP,BIP1,BIM1,DP)
C
```

```
C NOW SOLVE THE LINEAR SYSTEMS OF EQ(14) FOR THIS I
C
      CALL SOLVIT(MM2,S(2,I),DS,BP,BIP1,BIM1,DP)
   40 CONTINUE
C
C
      RETURN
C
  900 PRINT 903,MAX,MAXS
      STOP
  905 PRINT 906,N,NMIN
      STOP
  907 PRINT 908,M,MMIN
      STOP
  911 PRINT 915,(X(I),I=1,N)
      STOP
  912 PRINT 916,(Y(J),J=1,M)
      STOP
C
C --------------FORMAT STATEMENTS ---------------------------------
C
  903 FORMAT(10X,*ERROR*THE PARAMETER MAX OF SUB, BICUB1 HAS SET TO *,
     X I5,*, IT SHOULD BE *,I5)
  906 FORMAT(10X,*ERROR*THE X VECTOR HAS *,I3,
     X *POINTS AND THE MINIMUM ALLOWED IS *,I3)
  908 FORMAT(10X,*ERROR*THE Y VECTOR HAS *,I3,
     X *POINTS AND THE MINIMUM ALLOWED IS *,I3)
  915 FORMAT(10X,*ERROR*THE X VECTOR IS NOT ARRANGED PROPERLY*,//,
     X 10X,*ERROR DETECTED BY BICUB1*,/,
     X 10X,*THE X VECTOR IS*,/,
     X(5(2X,F12,4)))
  916 FORMAT(10X,*ERROR*THE Y VECTOR IS NOT ARRANGED PROPERLY*,//,
     X 10X,*ERROR DETECTED BY BICUB1*,/,
     X 10X,*THE Y VECTOR IS*,/,
     X(5(2X,F12,4)))
C
      END
```

```
      SUBROUTINE BICUB2(XPT,YPT,UPT,NERROR,N,M,X,Y,U,P,Q,S)
C
C SUBROUTINE TO INTERPOLATE A VALUE,UPT, AT LOCATION (XPT,YPT),
C THIS SUBROUTINE ASSUMES THAT THE X AND Y VECTORS ARE LARGE ENOUGH
C TO MAKE A BINARY SEARCH TECHNIQUE SUPERIOR TO A
C SEQUENTIAL SEARCH TECHNIQUE.
C NERROR IS SET TO 1 IF (XPT,YPT) IS OUTSIDE THE PROPER DOMAIN
C
      DIMENSION X(N),Y(M),U(N,M),P(N,M),Q(M,N),S(M,N)
C
      NERROR=0
C
C CONDUCT A BINARY SEARCH FOR I
C
      KH=N
      KL=1
    2 I=(KL+KH)/2
    3 IF(X(I)-XPT) 6,7,11
   11 IF(XPT-X(I-1)) 4,18,7
    4 KH=I
    5 IF(KH-KL-1) 9,9,2
    6 KL=I
      GO TO 5
    9 IF(XPT-X(KH)) 10,8,13
    8 I=KH
      GO TO 7
   10 IF(XPT-X(KL)) 13,14,8
   14 I=KL
      GO TO 7
   18 I=I-1
      GO TO 7
   13 NERROR=1
      PRINT 12
      UPT=0.0
      RETURN
C
C CONDUCT A BINARY SEARCH FOR J
C
    7 KH=M
      KL=1
   20 J=(KL+KH)/2
   30 IF(Y(J)-YPT) 60,70,110
  110 IF(YPT-Y(J-1)) 40,180,70
   40 KH=J
   50 IF(KH-KL-1)90,90,20
   60 KL=J
      GO TO 50
   90 IF(YPT-Y(KH))  100,80,130
   80 J=KH
      GO TO 70
  100 IF(YPT-Y(KL)) 130,140,80
  140 J=KL
      GO TO 70
  180 J=J-1
      GO TO 70
  130 NERROR=1
```

```
      PRINT 23
      UPT=0.0
      RETURN
C
C
C USE A CUBIC HERMITE BASIS TO EVALUATE THE BICUBIC POLYNOMIAL
C AT (XPT,YPT) .
C     (XPT,YPT) LIES WITHIN THE RECTANGLE BOUNDED BY X(I),X(I-1),Y(J),
C AND Y(J-1).
C
   70 IM1=I-1
      JM1=J-1
      XI=X(I)
      YJ=Y(J)
      XIM1=X(IM1)
      YJM1=Y(JM1)
      XMX1OH=(XPT-XIM1)/(XI-XIM1)
      YMY1OK=(YPT-YJM1)/(YJ-YJM1)
      CX1=(3.-2.*XMX1OH)*XMX1OH**2
      CY1=(3.-2.*YMY1OK)*YMY1OK**2
      CX2=(XI-XPT)*XMX1OH
      CY2=(YJ-YPT)*YMY1OK
      CX3=XMX1OH*CX2
      CY3=YMY1OK*CY2
      CX2=CX2-CX3
      CY2=CY2-CY3
      UIJ1=U(I,JM1)
      UI1J1=U(IM1,JM1)
      PIJ1=P(I,JM1)
      PI1J1=P(IM1,JM1)
      TIM1=UI1J1+CY1*(U(IM1,J)-UI1J1)+CY2*Q(JM1,IM1)-CY3*Q(J,IM1)
      TI=UIJ1+CY1*(U(I,J)  -UIJ1)+CY2*Q(JM1,I)-CY3*Q(J,I)
      TPIM1=PI1J1+CY1*(P(IM1,J)-PI1J1)+CY2*S(JM1,IM1)-CY3*S(J,IM1)
      TPI=PIJ1+CY1*(P(I,J)-PIJ1)+CY2*S(JM1,I)-CY3*S(J,I)
      UPT=TIM1+CX1*(TI-TIM1)+CX2*TPIM1-CX3*TPI
C
      RETURN
C
C ********************************
C     FORMAT STATEMENTS
C ********************************
C
   12 FORMAT(10X,*ERROR-XPT OUT OF BOUNDS*,/,
     X 10X,*DETECTED BY SUB. BICUB2*)
   23 FORMAT(10X,*ERROR-YPT OUT OF BOUNDS*,/,
     X 10X,*DETECTED BY BICUB2*)
C
      END
```

```
      SUBROUTINE EDGES(N,M,X,Y,U,P,Q,S)
C
C IF THE USER REQUESTED  DEFAULT EDGE CONDITIONS
C THIS SUBROUTINE WILL DETERMINE THEM
C USING A LAGRANGE INTERPOLATING POLYNOMIAL OF ORDER 3 X 3
C
C
C THIS SUBROUTINE IS CALLED BY SUBROUTINE BICUR1.
C THIS SUBROUTINE CALLS SUBROUTINE LAGRAN.
C
      DIMENSION X(N),Y(M),U(N,M),P(N,M),Q(M,N),S(M,N)
C
C
C GET PARTIALS WITH RESPECT TO X ALONG EDGES
C
C
C DETERMINE THE LOCATION OF THE 4 X 4 GRID FOR THE LAGRANGE POLYNOMIAL
C
      DO 10 J=1,M
      MS=J-2
      IF(MS .GT. (M-3)) MS=M-3
      IF(MS .LT. 1) MS=1
      MF=MS+3
      CALL LAGRAN(2,1,4,MS,MF,P(1,J),1,J,N,M,X,Y,U,P,Q,S)
      CALL LAGRAN(2,N-3,N,MS,MF,P(N,J),N,J,N,M,X,Y,U,P,Q,S)
   10 CONTINUE
C
C GET PARTIALS WITH RESPECT TO Y ALONG EDGES
C
      DO 20 I=1,N
      NS=I-2
      IF(NS .GT.(N-3)) NS=N-3
      IF(NS .LT. 1) NS=1
      NF=NS+3
      CALL LAGRAN(3,NS,NF,1,4,Q(1,I),I,1,N,M,X,Y,U,P,Q,S)
      CALL LAGRAN(3,NS,NF,M-3,M,Q(M,I),I,M,N,M,X,Y,U,P,Q,S)
   20 CONTINUE
C
C GET PARTIALS WITH RESPECT TO X AND Y AT CORNERS
C
      CALL LAGRAN(4,1,4,1,4,S(1,1),1,1,N,M,X,Y,U,P,Q,S)
      CALL LAGRAN(4,N-3,N,1,4,S(1,N),N,1,N,M,X,Y,U,P,Q,S)
      CALL LAGRAN(4,1,4,M-3,M,S(M,1),1,M,N,M,X,Y,U,P,Q,S)
      CALL LAGRAN(4,N-3,N,M-3,M,S(M,N),N,M,N,M,X,Y,U,P,Q,S)
      RETURN
      END
```

```
      SUBROUTINE GETBP(N,V,BP,BIP1,BIM1)
C
C SUBROUTINE TO GET THE 2 DIMENSIONAL DIFFERENCE ARRAY,B, OF EQ(15) OF
C REFERENCE AND ALSO THE B PRIME ARRAY FOR GAUSSIAN ELIMINATION
C
C THIS SUBROUTINE IS CALLED BY SUBROUTINE BICUB1.
C
C
C TO SAVE STORAGE WE
C STORE ELEMENT B(I,I) IN BP(I)
C STORE ELEMENT B(I,I+1) IN BIP1(I)
C STORE ELEMENT B(I,I-1) IN BIM1(I)
C THIS REDUCES THE SPACE REQUIRED FOR THE B AND B PRIME ARRAYS
C FROM 2*N**2 TO 3*N WORDS
C
      DIMENSION BP(N),BIP1(N),BIM1(N),V(N)
C
      NM1=N-1
      NM2=N-2
      NM3=N-3
      DXR=V(2)-V(1)
      BP(1)=2.*(DXR+(V(3)-V(2)))
      BIP1(1)=DXR
      DXL=V(N)-V(NM1)
      BIM1(NM2)=DXL
      BP(NM2)=2.*(DXL+(V(NM1)-V(NM2)))
      IF(N .EQ. 4) GO TO 11
C
      DO 10 I=2,NM3
      IM1=I-1
      IP1=I+1
      IP2=IP1+1
      DXL=V(IP2)-V(IP1)
      DXR=V(IP1)-V(I)
      BIM1(I)=DXL
      BP(I)=2.*(DXL+DXR)
      BIP1(I)=DXR
   10 CONTINUE
C
C
C NOW DETERMINE THE B PRIME MATRIX
C
   11 DO 20 I=2,NM2
      IM1=I-1
      BP(I)=BP(I)-BIM1(I)*BIP1(IM1)/BP(IM1)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE LAGRAN(NTYPE,NS,NF,MS,MF,DPT,NPT,MPT,N,M,X,Y,U,P,Q,S)
C
C
C SUBROUTINE FOR INTERPOLATING THE VALUE OF A FUNCTION AND ITS
C DERIVATIVES WITH RESPECT TO X, WITH RESPECT TO Y, AND WITH RESPECT TO
C X AND Y AT THE MESH POINTS USING A LAGRANGE INTERPOLATING
C POLYNOMIAL OF ARBITRARY ORDER
C
C THIS SUBROUTINE IS CALLED BY SUBROUTINE EDGES.
C
C
C NS= NUMBER OF STARTING POINT ALONG THE X XAIS
C NF= FINAL POINT ALONG THE X AXIS
C MS= STARTING POINT ALONG THE Y AXIS
C MF= FINAL POINT ALONG THE Y AXIS
C INTERPOLATION IS CARRIED OUT OVER THESE POINTS
C
C
C SUBROUTINE LAGRAN HAS 16 PARAMETERS IN ITS CALLING SEQUENCE
C
C
C NTYPE=1  GET FUNCTION ITSELF
C NTYPE=2 GET PARTIAL WITH RESPECT TO X
C NTYPE =3 GET PARTIAL WITH RESPECT TO Y
C NTYPE=4 GET PARTIAL WITH RESPECT TO X AND Y
C
C
      DIMENSION X(N),Y(M),U(N,M),P(N,M),Q(M,N),S(M,N)
C
      XP=X(NPT)
      YP=Y(MPT)
      DPT=0.
      GO TO(10,20,30,40),NTYPE
C
C CALCULATE THE VALUE OF THE FUNCTION AT THE MESH POINT (NPT,MPT)
C
   10 DPT=U(NPT,MPT)
      RETURN
C
C
C CALCULATE THE FIRST PARTIAL WITH RESPECT TO X
C AT THE MESH POINT (NPT,MPT)
C
   20 DO 211 I=NS,NF
      SUM=0.
      DO 215 L=NS,NF
      IF(L.EQ. I) GO TO 215
      PROD=1.
      DO 212 K=NS,NF
      IF(K.EQ. I .OR. K.EQ. L  ) GO TO 212
      PROD=PROD*(XP-X(K))
  212 CONTINUE
      SUM=SUM+PROD
  215 CONTINUE
      DENOM=1.
      XI=X(I)
```

```
      DO 213 K=NS,NF
      IF(K .EQ. I) GO TO 213
      DENOM=DENOM*(XI-X(K))
  213 CONTINUE
      DPT=DPT+SUM/DENOM*L(I,MPT)
  211 CONTINUE
      RETURN
C
C CALCULATE THE FIRST PARTIAL WITH RESPECT TO Y
C AT THE MESH POINT (NPT,MPT)
C
   30 DO 311 J=MS,MF
      SUM=0.
      DO 315 L=MS,MF
      IF(L .EQ. J) GO TO 315
      PROD=1.
      DO 312 K=MS,MF
      IF(K.EQ. J .OR. K.EQ. L  ) GO TO 312
      PROD=PROD*(YP-Y(K))
  312 CONTINUE
      SUM=SUM+PROD
  315 CONTINUE
      DENOM=1.
      YJ=Y(J)
      DO 313 K=MS,MF
      IF(K.EQ. J) GO TO 313
      DENOM=DENOM*(YJ-Y(K))
  313 CONTINUE
      DPT=DPT+SUM/DENOM*L(NPT,J)
  311 CONTINUE
      RETURN
C
C  CALCULATE THE PARTIAL WITH RESPECT TO X AND Y
C AT THE MESH POINT (NPT,MPT)
C
   40 DO 41 I=NS,NF
      SUMX=0.
      DO 43 L=NS,NF
      IF(L.EQ. I) GO TO 43
      PROD=1.
      DO 44 K=NS,NF
      IF(K.EQ. I .OR. K.EQ. L) GO TO 44
      PROD=PROD*(XP-X(K))
   44 CONTINUE
      SUMX=SUMX+PROD
   43 CONTINUE
      DENOM=1.
      XI=X(I)
      DO 45 K=NS,NF
      IF(K .EQ.I) GO TO 45
      DENOM=DENOM*(XI-X(K))
   45 CONTINUE
      SUMX=SUMX/DENOM
      DO 42 J=MS,MF
      SUMY=0.
      DO 433 L=MS,MF
```

```
      IF(L.EQ. J) GO TO 433
      PROD=1.
      DO 434 K=MS,MF
      IF(K.EQ. J.OR. K.EQ. L) GO TO 434
      PROD=PROD*(YP-Y(K))
  434 CONTINUE
      SUMY=SUMY+PROD
  433 CONTINUE
      DENOM=1.
      YJ=Y(J)
      DO 435 K=MS,MF
      IF(K .EQ. J) GO TO 435
      DENOM=DENOM*(YJ-Y(K))
  435 CONTINUE
      SUMY=SUMY/DENOM
      DPT=DPT+SUMY*SUMX*L(I,J)
   42 CONTINUE
   41 CONTINUE
      RETURN
C
      END
```

```
      SUBROUTINE SOLVIT(N,Z,D,BP,BIP1,BIM1,DP)
C
C
C THIS ROUTINE IS CALLED BY SUBROUTINE BICUB1
C
      DIMENSION BP(N),BIP1(N),BIM1(N),D(N),Z(N),DP(N)
C
C COMPUTE THE D PRIME VECTOR, SEE EQ(16) OF REF
C
      DP(1)=D(1)
      DO 10 I=2,N
      IM1=I-1
      DP(I)=D(I)-BIM1(I)*DP(IM1)/BP(IM1)
   10 CONTINUE
C
C OBTAIN SOLUTION BY RECURSION RELATION OF EQ(17) , SEE REF
C
      Z(N)=DP(N)/BP(N)
      NM1=N-1
      DO 20 J=1,NM1
      I=N-J
      IP1=I+1
      Z(I)=(DP(I)-BIP1(I)*Z(IP1))/BP(I)
   20 CONTINUE
      RETURN
      END
```

6.0 COMPARISONS

For most quantities defined over a two dimensional mesh, where N and M are greater than 4, the bicubic spline generates a more physically plausible interpolation surface than a two dimensional Lagrange interpolation polynomial over the same mesh. When N and M are equal to 4 the bicubic spline and Lagrange interpolating polynomial are identical.

No comparisons have been made with any other programs.

7.0 TEST METHODS AND RESULTS

The following program, CHECK, illustrates the use of the routines.

CHECK begins by setting up an arbitrary 5 x 6 mesh using a data statement to define the X and Y arrays. Next a third order two dimensional polynomial, U(I,J), having arbitrary coefficients is evaluated at each of the mesh points. Since NDFAULT is initially set to zero, CHECK is required to supply the normal derivatives along the boundaries. The equations used in statements 300, 301, and 302 were obtained by differentiating the polynomial U(I,J). Next subroutine BICUB1 is called to complete the P, Q, and S arrays and the results are printed. Following this, 30 random coordinates are generated over the x-y mesh using a uniform random number generator available in the CDC-3800 system library and the polynomial U(I,J) is evaluated at each point (UEXACT). Also subroutine BICUB2 is called to interpolate a value at each point. Then the x-y coordinates of the 30 points are printed out. Since the arbitrary polynomial is third order in x and y, the interpolated and exact values should be the same. The fact that the elements of the difference column are essentially zero (neglecting rounding errors) indicates that this is indeed the case. Next the P, Q, and S arrays are zeroed and the parameter NDFAULT is set to 1, indicating that BICUB1 should determine the edge conditions rather than assume they are supplied by CHECK, and the above procedure is repeated. Again the difference column is essentially zero. In addition, the P, Q, and S arrays determined by BICUB1 agree almost exactly with those obtained by CHECK.

```
      PROGRAM CHECK
C
C SAMPLE PROGRAM TO ILLUSTRATE THE USE OF THE
C BICUBIC SPLINE INTERPOLATION PACKAGE
C
C SETTING UP AN ARBITRARY MESH
C
      DATA(N=5),(M=6),(MAX=6)
      DATA(X=1.,2.5,2.75,3.,5.),(Y=1.,1.5,2.25,4.5,5.,7.3)
C
      DIMENSION X(5),Y(6),U(5,6),P(5,6),Q(6,5),S(6,5)
C
C SET THE DIMENSION OF EACH OF THE SEVEN WORK AREAS TO MAX.
C
      DIMENSION W1(6),W2(6),W3(6),W4(6),W5(6),W6(6),W7(6)
C
C EVALUATE AN ARBITRARY POLYNOMIAL AT EACH MESH POINT
C
      DO 12 I=1,N
      XI=X(I)
      DO 20 J=1,M
      YI=Y(J)
      U(I,J)=3.+15.*XI+17.*XI**2+83.*XI**3
     X+YI*(45.+26.*XI+18.*XI**2+19.*XI**3)
     X +YI**2*(34.+6.*XI+13.*XI**2+43.*XI**3)
     X  +YI**3*(47.+21.*XI+15.*XI**2+2.*XI**3)
   20 CONTINUE
   12 CONTINUE
C
C
C IF NDFAULT IS SET TO 0 THE EDGE CONDITIONS WILL BE INPUT TO BICUB1.
C IF NDFAULT IS SET TO 1 BICUB1 WILL CALCULATE THE EDGE CONDITIONS.
C
      NDFAULT=0
C
C CALCULATE EXACT EDGE CONDITIONS FOR TEST EQUATION
C
      NM1=N-1
      MM1=M-1
      DO 300 I=1,N,NM1
      XI=X(I)
C
C GET NORMAL DERIVATIVES WITH RESPECT TO X
C
      DO 300 J=1,M
      YI=Y(J)
  300 P(I,J)=15.+34.*XI+249.*XI**2
     X  +YI*(26.+36.*XI+57.*XI**2)
     X   +YI**2*(6.+26.*XI+129.*XI**2)
     X  +YI**3*(21.+30.*XI+6.*XI**2)
C

C GET NORMAL DERIVATIVES WITH RESPECT TO Y ALONG EDGE
C
      DO 301 J=1,M,MM1
      YI=Y(J)
```

```
      DO 301 I=1,N
      XI=X(I)
  301 Q(J,I)=45.+26.*XI+18.*XI**2+19.*XI**3
     X    +2.*YI*(34.+6.*XI+13.*XI**2+43.*XI**3)
     X    +3.*YI**2*(47.+21.*XI+15.*XI**2+2.*XI**3)
C
C GET NORMAL DERIVATIVES WITH RESPECT TO BOTH X AND Y AT EACH CORNER OF
C MESH.
C
      DO 302 I=1,N,NM1
      XI=X(I)
      DO 302 J=1,M,MM1
      YI=Y(J)
  302 S(J,I)=26.+36.*XI+57.*XI**2
     X     +2.*YI*(6.+26.*XI+129.*XI**2)
     X    +3.*YI**2*(21.+30.*XI+6.*XI**2)
C
C
C ESTIMATE THE AMOUNT OF TIME REQUIRED FOR A CALL TO BICUB1
C
  200 T=TIMELEFT(1)
C
C COMPLETE THE P,Q, AND S ARRAYS , THAT IS, DETERMINE NORMAL DERIVATIVES
C AT EACH MESH POINT.
C
C *****************************************************************
      CALL BICUB1(N,M,NDFALLT,X,Y,U,P,Q,S,MAX,W1,W2,W3,W4,W5,W6,W7)
C *****************************************************************
C
C
C CALCULATE TIME SPENT IN MILLISECONDS
C
      T=(T-TIMELEFT(1))*1000.
      PRINT 3
      DO 100 I=1,N
      DO 110 J=1,M
      PRINT 111,I,J,X(I),Y(J),U(I,J),P(I,J),Q(J,I),S(J,I)
  110 CONTINUE
  100 CONTINUE
      PRINT 4,T
      PRINT 2
C
C USE SYSTEM RANDOM NUMBER GENERATOR TO
C GENERATE RANDOM COORDINATES OVER THE X-Y PLANE
C
      DO 10 K=1,30
      XI=RANF(-1)*(X(N)-X(1))+X(1)-.3*RANF(-1)
      YL=RANF(-1)*(Y(M)-Y(1))+Y(1)-.3*RANF(-1)
      UEXACT=3.+15.*XI+17.*XI**2+83.*XI**3
     X+YI*(45.+26.*XI+18.*XI**2+19.*XI**3)
     X +YI**2*(34.+6.*XI+13.*XI**2+43.*XI**3)
     X  +YI**3*(47.+21.*XI+15.*XI**2+2.*XI**3)
C
C
C ESTIMATE THE AMOUNT OF TIME REQUIRED FOR THE CALL TO BICUB2
C
```

FTN5.5A

```
      T=TIMELEFT(1)
C INTERPOLATE AT VALUE AT (XI,YI)
C
C ***********************************************************************
      CALL BICUB2(XI,YI,UINT,NERROR,N,M,X,Y,U,P,Q,S)
C ***********************************************************************
C
      T=(T-TIMELEFT(1))*1000.
      DIFF=UINT-UEXACT
      PRINT 1,K,XI,YI,UEXACT,UINT,DIFF,NERROR,T
   10 CONTINUE
C
C SEE IF WE HAVE COMPLETED THE SECOND PASS
C
      IF(NDFAULT .EQ. 1) STOP
C
C REDEFINE NDFAULT SO THAT THE EDGE CONDITIONS WILL BE  CALCULATED
C AND ZERO OUT THE P,Q, AND S ARRAYS.
C
      NDFAULT=1
      DO 501 J=1,M
      DO 500 I=1,N
      P(I,J)=Q(J,I)=S(J,I)=0.
  500 CONTINUE
  501 CONTINUE

C REPEAT ASSUMING BOUNDARY CONDITIONS ARE NOT GIVEN.
C HERE WE USE THE LAGRANGE INTERPOLATING POLYNOMIAL TO DETERMINE THE
C EDGE CONDITIONS.
C
      GO TO 200
C
C FORMAT STATEMENTS ************
C
    1 FORMAT(10X,I5,2X,4(F12.4,2X),E16.5,2X,I5,2X,F12.4)
    2 FORMAT(1X,//,
     X 14X,*K*,9X,*XI*,12X,*YI*,9X,*UEXACT*,8X,*UINT*,12X,
     X *DIFFERENCE*,5X,*NERROR*,5X,*TIME*,/)
    3 FORMAT(1X,//,3X,*I*,3X,*J*,9X,*X*,12X,*Y*,12X,*U*,12X,*P*,12X,*Q*,
     X 12X,*S*,/)
    4 FORMAT(10X,*THE CALL TO BICUB1 TOOK APPROX.*,F12.4,* MSEC.*)
  111 FORMAT(1X,2(I3,1X),6(E12.4,1X))
C
      END
```

I	J	X	Y	U	P	Q	S
1	1	1.0000+000	1.0000+000	4.0700+002	6.3500+002	5.5500+002	6.1200+002
1	2	1.0000+000	1.5000+000	7.8287+002	1.0311+003	9.6975+002	9.8675+002
1	3	1.0000+000	2.2500+000	1.8152+003	2.0301+003	1.8309+003	1.7092+003
1	4	1.0000+000	4.5000+000	1.0294+004	9.2879+003	6.1358+003	5.0308+003
1	5	1.0000+000	5.0000+000	1.3683+004	1.2043+004	7.4430+003	6.0040+003
1	6	1.0000+000	7.3000+000	3.9089+004	3.1920+004	1.5099+004	1.1582+004
2	1	2.5000+000	1.0000+000	2.9896+003	3.1393+003	2.7971+003	2.6273+003
2	2	2.5000+000	1.5000+000	4.7852+003	4.7890+003	4.4411+003	4.0051+003
2	3	2.5000+000	2.2500+000	9.2302+003	8.6805+003	7.5385+003	6.4474+003
2	4	2.5000+000	4.5000+000	4.0481+004	3.3711+004	2.1377+004	1.6478+004
2	5	2.5000+000	5.0000+000	5.2156+004	4.2636+004	2.5378+004	1.9257+004
2	6	2.5000+000	7.3000+000	1.3531+005	1.0379+005	4.8121+004	3.4673+004
3	1	2.7500+000	1.0000+000	3.8496+003	3.7496+003	3.5133+003	3.1088+003
3	2	2.7500+000	1.5000+000	6.0943+003	5.6975+003	5.5305+003	4.7202+003
3	3	2.7500+000	2.2500+000	1.1596+004	1.0270+004	9.2870+003	7.5559+003
3	4	2.7500+000	4.5000+000	4.9609+004	3.9385+004	2.5817+004	1.9078+004
3	5	2.7500+000	5.0000+000	6.3687+004	4.9708+004	3.0562+004	2.2252+004
3	6	2.7500+000	7.3000+000	1.6327+005	1.2008+005	5.7408+004	3.9731+004
4	1	3.0000+000	1.0000+000	4.8690+003	4.4150+003	4.3550+003	3.6320+003
4	2	3.0000+000	1.5000+000	7.6406+003	6.6866+003	6.8063+003	5.4958+003
4	3	3.0000+000	2.2500+000	1.4376+004	1.1996+004	1.1324+004	8.7554+003
4	4	3.0000+000	4.5000+000	6.0212+004	4.5516+004	3.0932+004	2.1876+004
4	5	3.0000+000	5.0000+000	7.7057+004	5.7343+004	3.6523+004	2.5472+004
4	6	3.0000+000	7.3000+000	1.9546+005	1.3761+005	6.8017+004	4.5203+004
5	1	5.0000+000	1.0000+000	2.0419+004	1.1723+004	1.6859+004	9.3160+003
5	2	5.0000+000	1.5000+000	3.0969+004	1.7502+004	2.5537+004	1.3881+004
5	3	5.0000+000	2.2500+000	5.5659+004	3.0751+004	4.0739+004	2.1631+004
5	4	5.0000+000	4.5000+000	2.1190+005	1.1106+005	1.0208+005	5.1381+004
5	5	5.0000+000	5.0000+000	2.6710+005	1.3871+005	1.1891+005	5.9316+004
5	6	5.0000+000	7.3000+000	6.4221+005	3.2230+005	2.1137+005	1.0202+005

THE CALL TO BICUB1 TOOK APPROX. 23.0000 MSEC.

K	XI	YI	UEXACT	UINT	DIFFERENCE	NERROR	TIME
1	4.6801	7.3000	544727.7631	544727.7631	0.00000+000	0	3.0000
2	3.9544	7.3000	362712.4656	362712.4656	7.62939-006	0	2.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
3	0.9417	7.3000	37283.9349	-0.0000	-3.72839+004	1	15.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
4	5.1564	7.3000	693996.9885	-0.0000	-6.93997+005	1	14.0000
5	2.4001	7.3000	125257.4696	125257.4696	-1.90735-006	0	2.0000
6	4.7003	7.3000	550557.2826	550557.2826	1.52588-005	0	3.0000
7	3.8947	7.3000	349989.5102	349989.5102	0.00000+000	0	3.0000
8	3.2720	7.3000	235644.8094	235644.8094	0.00000+000	0	3.0000
9	2.3545	7.3000	120867.6676	120867.6676	-3.81470-006	0	4.0000
10	3.4317	7.3000	261904.0295	261904.0295	3.81470-006	0	2.0000
11	4.6961	7.3000	549341.0577	549341.0577	-3.05176-005	0	2.0000
12	1.5016	7.3000	59680.5639	59680.5639	0.00000+000	0	2.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
13	0.9623	7.3000	37908.1032	-0.0000	-3.79081+004	1	14.0000
14	2.7008	7.3000	157443.8451	157443.8451	-3.81470-006	0	3.0000
15	2.9853	7.3000	193437.7735	193437.7735	0.00000+000	0	2.0000
16	4.9281	7.3000	619325.4995	619325.4995	-1.52588-005	0	3.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
17	5.4112	7.3000	784423.6679	-0.0000	-7.84424+005	1	14.0000
18	3.4676	7.3000	268078.4195	268078.4195	0.00000+000	0	4.0000
19	2.5798	7.3000	143800.8440	143800.8440	-3.81470-006	0	2.0000
20	2.0962	7.3000	95141.0646	95141.0646	3.81470-006	0	3.0000
21	4.0475	7.3000	383210.2274	383210.2274	0.00000+000	0	3.0000
22	4.3128	7.3000	446125.0424	446125.0424	7.62939-006	0	4.0000
23	1.6377	7.3000	67029.4197	67029.4197	1.90735-006	0	2.0000
24	3.2053	7.3000	225272.4446	225272.4446	-3.81470-006	0	3.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
25	5.5974	7.3000	855451.8719	-0.0000	-8.55452+005	1	13.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
26	5.3811	7.3000	773367.9365	-0.0000	-7.73368+005	1	14.0000
27	1.1284	7.3000	43467.2707	43467.2707	0.00000+000	0	4.0000
28	2.7850	7.3000	167520.5473	167520.5472	-3.81470-006	0	4.0000
29	4.3106	7.3000	445564.0880	445564.0880	7.62939-006	0	3.0000
30	2.6334	7.3000	149722.1129	149722.1129	-3.81470-006	0	2.0000

I	J	X	Y	U	P	Q	S
1	1	1.0000+000	1.0000+000	4.0700+002	6.3500+002	5.5500+002	6.1200+002
1	2	1.0000+000	1.5000+000	7.8287+002	1.0311+003	9.6975+002	9.8675+002
1	3	1.0000+000	2.2500+000	1.8152+003	2.0301+003	1.8309+003	1.7092+003
1	4	1.0000+000	4.5000+000	1.0294+004	9.2879+003	6.1358+003	5.0308+003
1	5	1.0000+000	5.0000+000	1.3683+004	1.2043+004	7.4430+003	6.0040+003
1	6	1.0000+000	7.3000+000	3.9089+004	3.1920+004	1.5099+004	1.1582+004
2	1	2.5000+000	1.0000+000	2.9896+003	3.1393+003	2.7971+003	2.6272+003
2	2	2.5000+000	1.5000+000	4.7852+003	4.7890+003	4.4411+003	4.0051+003
2	3	2.5000+000	2.2500+000	9.2302+003	8.6805+003	7.5385+003	6.4474+003
2	4	2.5000+000	4.5000+000	4.0481+004	3.3711+004	2.1377+004	1.6478+004
2	5	2.5000+000	5.0000+000	5.2156+004	4.2636+004	2.5378+004	1.9257+004
2	6	2.5000+000	7.3000+000	1.3531+005	1.0379+005	4.8121+004	3.4623+004
3	1	2.7500+000	1.0000+000	3.8496+003	3.7496+003	3.5133+003	3.1088+003
3	2	2.7500+000	1.5000+000	6.0943+003	5.6975+003	5.5305+003	4.7202+003
3	3	2.7500+000	2.2500+000	1.1596+004	1.0270+004	9.2870+003	7.5559+003
3	4	2.7500+000	4.5000+000	4.9609+004	3.9385+004	2.5817+004	1.9078+004
3	5	2.7500+000	5.0000+000	6.3687+004	4.9708+004	3.0562+004	2.2252+004
3	6	2.7500+000	7.3000+000	1.6327+005	1.2008+005	5.7408+004	3.9731+004
4	1	3.0000+000	1.0000+000	4.8690+003	4.4150+003	4.3550+003	3.6320+003
4	2	3.0000+000	1.5000+000	7.6406+003	6.6866+003	6.8063+003	5.4958+003
4	3	3.0000+000	2.2500+000	1.4376+004	1.1996+004	1.1324+004	8.7554+003
4	4	3.0000+000	4.5000+000	6.0212+004	4.5516+004	3.0932+004	2.1876+004
4	5	3.0000+000	5.0000+000	7.7057+004	5.7343+004	3.6523+004	2.5472+004
4	6	3.0000+000	7.3000+000	1.9546+005	1.3761+005	6.8017+004	4.5203+004
5	1	5.0000+000	1.0000+000	2.0419+004	1.1723+004	1.6859+004	9.3160+003
5	2	5.0000+000	1.5000+000	3.0969+004	1.7502+004	2.5537+004	1.3881+004
5	3	5.0000+000	2.2500+000	5.5659+004	3.0751+004	4.0739+004	2.1631+004
5	4	5.0000+000	4.5000+000	2.1190+005	1.1106+005	1.0208+005	5.1381+004
5	5	5.0000+000	5.0000+000	2.6710+005	1.3871+005	1.1891+005	5.9316+004
5	6	5.0000+000	7.3000+000	6.4221+005	3.2230+005	2.1137+005	1.0202+005

THE CALL TO BICUB1 TOOK APPROX. 135.0000 MSEC.

K	XI	YI	UEXACT	UINT	DIFFERENCE	NERROR	TIME
1	3.8803	7.3000	346979.9898	346979.9899	9.15527-005	0	3.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
2	0.7365	7.3000	31767.4360	-0.0000	-3.17674+004	1	14.0000
3	3.7808	7.3000	326597.6141	326597.6142	6.86644-005	0	3.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
4	5.2111	7.3000	712790.2950	-0.0000	-7.12790+005	1	14.0000
5	2.7605	7.3000	164541.3793	164541.3793	-3.81470-006	0	2.0000
6	1.2772	7.3000	49278.4133	49278.4134	2.57492-005	0	4.0000
7	4.1136	7.3000	398260.6176	398260.6177	9.15527-005	0	3.0000
8	2.1946	7.3000	106446.5736	106446.5736	1.14441-005	0	4.0000
9	4.0813	7.3000	390873.5700	390873.5701	9.91821-005	0	3.0000
10	1.1850	7.3000	45579.5901	45579.5901	1.81198-005	0	2.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
11	5.2737	7.3000	734675.3059	-0.0000	-7.34675+005	1	14.0000
12	4.7600	7.3000	568042.4312	568042.4313	7.62939-005	0	4.0000
13	1.1182	7.3000	43096.8970	43096.8970	1.33514-005	0	2.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
14	5.2122	7.3000	713147.0529	-0.0000	-7.13147+005	1	14.0000
15	3.8593	7.3000	342596.3454	342596.3455	9.91821-005	0	3.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
16	0.7218	7.3000	31417.4948	-0.0000	-3.14175+004	1	14.0000
17	2.6614	7.3000	152901.5849	152901.5849	0.00000+000	0	4.0000
18	2.8192	7.3000	171750.5687	171750.5687	0.00000+000	0	2.0000
19	1.2153	7.3000	46760.5225	46760.5225	1.90735-005	0	2.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
20	5.6675	7.3000	883290.6428	-0.0000	-8.83291+005	1	14.0000
21	2.8152	7.3000	171242.4343	171242.4343	-3.81470-006	0	2.0000
22	3.2704	7.3000	235392.1745	235392.1746	1.52588-005	0	4.0000
23	2.7102	7.3000	158550.9106	158550.9106	0.00000+000	0	3.0000
24	3.3746	7.3000	252282.3114	252282.3114	2.67029-005	0	4.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
25	0.7928	7.3000	33154.9680	-0.0000	-3.31550+004	1	14.0000
ERROR-XPT OUT OF BOUNDS DETECTED BY SUB. BICUB2							
26	5.2357	7.3000	721317.0978	-0.0000	-7.21317+005	1	15.0000
27	1.1668	7.3000	44889.5823	44889.5823	1.62125-005	0	2.0000
28	2.9207	7.3000	184767.6106	184767.6106	0.00000+000	0	3.0000
29	3.2510	7.3000	232340.2660	232340.2660	1.14441-005	0	4.0000
30	2.3937	7.3000	124634.3653	124634.3653	1.90735-006	0	2.0000

8.0 REMARKS

Although the Lagrange interpolation procedure described above for obtaining the "required" boundaries derivatives has been implemented and works, a different and possibly better approach [9] to this problem consists, in the analogous one dimensional case, of not having a break-point [6] at the second and second to last data points. Implementation of this latter approach is left as an exercise for the interested reader.

For those readers who may be interested, a completely independently conceived and different set of ALGØL procedures for bicubic spline interpolation is given in references [10 and 11] (in German). Comparison of Späth's algorithms with those described earlier is left as another exercise.

As a further remark, it should be obvious that the procedure described in this report could be readily generalized to N - dimensional cubic spline interpolation, where N is greater than 2.

9.0 ACKNOWLEDGMENTS

We are indebted to Professor Carl de Boor, of the University of Wisconsin, for reviewing an earlier draft of this report and offering a number of beneficial suggestions.

Also, we would like to thank Mrs. Janet Mason, of the NRL Research Computation Center, for checking the program and providing some helpful suggestions.